**Prog 260 Course Project**

This project is worth 20% of your grade.

**Requirements**

You will create a Windows Forms project that lets a user enter and store book information which will be stored in a Binary Search Tree. For each book, the Title, Author, Rating (1 poor, to 5 great) and PubYear (year published) will be stored and retrievable by using the books ISBN number.

Your form should have 4 buttons that allow:

* Entry of a new book object, which is created with the correct information and then stored into the BST (5 points)
* Entering an ISBN and retrieving an existing book’s information (with appropriate error message if there is no such book) (5 points)
* Entering an ISBN and removing (deleting) an existing book (5 points)
* Displaying a list of all the ISBN numbers currently in the BST (5 points)

Each of these 4 actions will rely on one method in the BST Class, 3 existing ones (that will need to be modified to store books) and one new one (to return a List of ISBNs) you must write.

**Suggestions**

I suggest you start examine both the BST Remove project that I posted, **4-RemoveB.zip** as well as your HW-6 Store Books in a BST project. Between these two projects you have most of the working code you need. You can start with either one, and copy parts from one into the other. If it was me, I would start with the working homework and then copy over the Remove method from the **4-RemoveB.zip** project. Many parts of these instructions are already done if you use the code fromm those 2 projects.

You will need a Book Class to store the book information required by this project. It needs to be extended just a bit from your homework to add a PubYear property. Note: In your BST class, that code will “think of” the ISBN number as the bstKey, while that same integer value over in in your winforms program will consider that bstKey to be the ISBN number. So, the form code will get an ISBN number from the user, and pass it into the BST Add method, where the Add method will use that passed in integer as the bstkey. Likewise, if a list of bstkey values is returned to the form, the form will consider it a list of ISBN numbers.

You will have to modify ALL the methods in the BST code to deal with this new added node property of type Book, so that each node holds a book object. Do not extend the BSTnode definition to store the Title, Author, Rating and PubYear directly. It should store a Book **object**, which in turn has those 4 items as properties. You will pass a **Book object** with an **ISBN (key)** from the form object to the BST class when you do an Add.

My suggestion for dealing with a request for a book that does not exist: Above I said you need to deal with the error when they request a book that does not exist. Here is how I did it:

* Have your book class definition set a new book object’s properties to “deliver a message” if there is no such book. For example, I used Title = "No such book", Author = "", Rating = 0, and YearPub = 0. If the user asks for a book that does not exist, I create this “empty book” and return it, so then the normal form code displays those values to alert the user there is no such book. If they ask to remove a book that does not exist, I just return, and no not indicate an error.

**Pay particular attention** to the Remove method. Since the Remove method does not return a book object, it is not to hard to modify it to deal with Books. When you are removing a node from the tree that has no or one child, there are no changes required for the Remove code. BUT, when you removed a node that has TWO children, the remove process copies an existing node up to a node to be removed. For this situation you will have to copy **both the key value AND the Book object.** That operation will occur 4 times in your project. **Please find all 4 places** where the remove code copies one node’s value’s into another and make that change. (If you don’t follow through on this warning, it will hurt your grade considerably, you will lose 2 points for any of the 4 places you miss.)

You should not have to change any of the pointer walking code, as you are just adding an object to the nodes, and not changing how the BST left and right pointers work.

You will have to write a new method that somehow returns all the key values in the BST (which over in the form code will be considered ISBN numbers). There are several ways this can be done. You might think of creating a List<int> in the new BSD method and return the List. However a List<int> is not so easy to display. It would be easier to create a new Class, ISBNclass, that has just one property, an int, called ISBN. Then copy and paste your Print BST code, and rename it to be GetISBNs(). Modify that code so that instead of writing the key value out to the console, it instead creates a new ISBNclass object, sets the ISBN property to the current key value, and then adds that new ISBNclass object to a List<ISBNclass> When you finish walking the tree in this new method, return your List to the form code.

Your form code will then display the List. To display the List<int> I suggest you drag a datagridview, which is in the toolbox menu under the Data subset, into your form.

Here I create some test code to create such a List just to show you how to display a List in a form:

List<ISBNclass> myList = new List<ISBNclass>();

ISBNclass item1 = new ISBNclass();

item1.ISBN = 23;

ISBNclass item2 = new ISBNclass();

Item2.ISBN = 42;

myList.Add(item1);

myList.Add(item2); // now I have a list with 2 objects in it

then, if that list exists, to display it in a Windows Form datagridview takes just one line of code:

dataGridView1.DataSource = myList;

Your BST class should have **no** Console.Write( xxxx) lines of code. Only the Form code is allowed to communicate with the user.

See the image below for what my form looks like. Yours can be anything you like, as long as it supports all the functionality required.

